

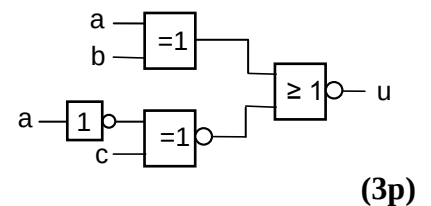
**TENTAMEN**

KURSNAMN	Digital- och datorteknik
PROGRAM:	Mekatronikingenjör (samt data- och elektroingenjör) Åk 1/ lp 2
KURSBETECKNING	LEU431
EXAMINATOR	Lars-Eric Arebrink
TID FÖR TENTAMEN	2010-12-13 kl 8.30 – 12.30
HJÄLPMEDEL	Av institutionen utgiven ”Instruktionslista för FLEX-processorn” (INS1) Tabellverk eller miniräknare får ej användas.
ANSV LÄRARE: Besöker tentamen	Lars-Eric Arebrink, tel. 772 5718 vid flera tillfällen
ANSLAG AV RESULTAT	Resultatlistor anslås senast 2010-12-20 på kursens hemsida. Granskning av rättning på institutionen 2010-12-20 och 2010-12-21 kl 9.00-9.30.
ÖVRIG INFORM. BETYGSGRÄNSER. SLUTBETYG	Tentamen omfattar totalt 60 poäng. Onödigt komplicerade lösningar kan ge poängavdrag. Svar på uppgifter skall motiveras. Betyg 3: 24 poäng Betyg 4: 36 poäng Betyg 5: 48 poäng För slutbetyg 3, 4 eller 5 på kursen fordras betyg 3, 4 eller 5 på tentamen och godkända laborationer.

1. I uppgift a-e nedan används 8-bitars tal. $X = 01011010$ och $Y = 10010101$.

- a) Visa med penna och papper hur räkneoperationen $R = X - Y$ utförs med en 8-bitars ALU. (1p)
- b) Vilka värden får flaggbitarna N, Z, V och C vid räkneoperationen? (1p)
- c) Tolka bitmönstren R, X och Y som tal *utan* tecken och ange deras decimala motsvarighet. Är resultatet R korrekt? Hur kan man avgöra detta med hjälp av flaggbitarna? (1p)
- d) Tolka bitmönstren R, X och Y som tal *med* tecken och ange deras decimala motsvarighet. Är resultatet R korrekt? Hur kan man avgöra detta med hjälp av flaggbitarna? (1p)
- e) Tolka bitmönstren X och Y som tal *med* tecken och ge ett booleskt uttryck $f(N, Z, V, C)$ som har värdet 1 om och endast om $X \leq Y$. Flaggvärdena är de som gäller efter subtraktionen ovan. (2p)
- f) Det hexadecimala talet $C3DA7000_{16}$ representerar ett flyttal enligt IEEE-standard 754-1985 med 23 bitar av mantissan och 8 bitars karakteristika. Vilket decimalt tal motsvarar det? (2p)

g) Ge ett minimalt boolesk SP-uttryck för u i grindnätet till höger.



2. En boolesk funktion $f(a, b, c, d)$ har karnaughdiagrammet till höger.

Realisera funktionen med så få grindar som möjligt. NAND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas. Endast insignalerna a, b, c och d finns tillgängliga.

		cd			
		00	01	11	10
ab	00	1	0	1	0
	01	0	1	0	1
	11	0	1	1	0
	10	0	1	1	0

(5p)

3.

- a) Ett synkront sekvensnät skall ha en insignal x och en utsignal u. Utsignalen u skall ges värdet "1" under ett bitintervall för varje insignalsekvens som består av tre ettor följda av en nolla och två ettor hos x.

Exempel: $\sigma_x = \dots 01100111011101100111110111011\dots$

$\sigma_u = \dots 00?00000001000100000000010001\dots$

Utsignalen skall som i exemplet ges värdet "1" när den sista biten i en korrekt insignalsekvens anländer på x-ingången och behålla värdet "1" så länge x har kvar sitt värde under detta bitintervall.

Rita en tillståndsgraf för sekvensnätet. Hur många vippor skulle minst krävas vid realiseringen?

(3p+1p)

- b) Realisera en räknare med räknevillkoret x och räknesekvensen q_1q_0 :

$x = 0$: 00, 10, 01, 11, 00, ...

$x = 1$: 00, 11, 01, 10, 00, ...

T-vippor, NAND-grindar med valfritt antal ingångar, XOR-grindar och NOT-grindar får användas.

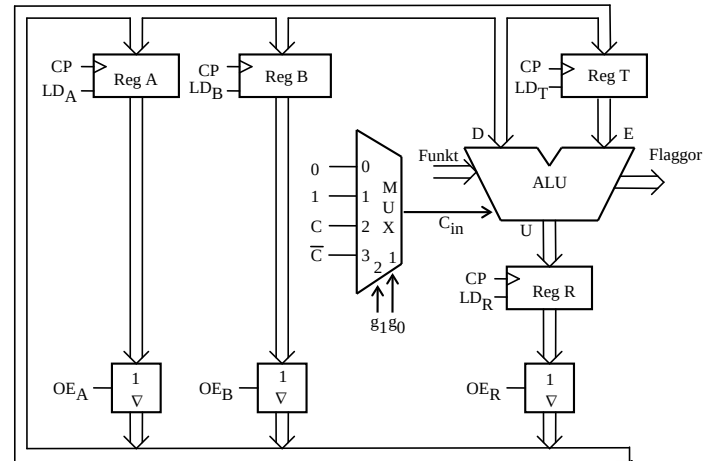
(5p)

4. Ge RTN-beskrivning och styrsignaler för de tillstånd krävs för att utföra operationen enligt nedanstående RTN-beskrivning:

RTN-beskrivning: $9 \cdot A - (B + 1) \rightarrow A$
(Aritmetisk multiplikation avses)

Använd den enkla datavägen till höger och ge ditt svar i tabellform.

Förutsätt att register A och B från början innehåller de data som skall behandlas enligt uttrycket ovan och att innehållen i register R och T är okända. Register B får inte ändras. Bortse från risken för overflow. Använd så få tillstånd som möjligt. Samtliga funktioner ALU:n kan utföra framgår av bilaga 1.



(5p)

5. Figur 1 i bilaga 3 visar hur datorn FLEX är uppbyggd. Bilaga 1 visar hur ALU'ns funktion väljs med styrsignalerna $f_3 - f_0$ och C_{in} .

I tabellen nedan visas styrsignalerna för de olika tillstånden i EXECUTE-sekvensen för en av FLEX-processorns instruktioner.

State	S-term	RTN-beskrivning	Styrsignaler (=1)
Q ₅	Q ₅ ·I _{xx}		OE _X , LD _T , DecSP
Q ₆	Q ₆ ·I _{xx}		OE _{SP} , LD _{MA}
Q ₇	Q ₇ ·I _{xx}		OE _{PC} , MW
Q ₈	Q ₈ ·I _{xx}		OE _A , f ₃ , f ₁ , LD _R
Q ₉	Q ₉ ·I _{xx}		OE _R , LD _{PC} , NF

- a) Rita en tabell med "State"- och RTN-kolumner enligt ovan och fyll i RTN-beskrivningen. Förklara vilken assemblerinstruktion som beskrivs. (2p)
- b) Instruktionen nedan skall implementeras för FLEX-processorn med hjälp av styrenheten med fast logik. Instruktionen består av tre ord och utför bitvis OR mellan dataordet som finns direkt efter operationskoden och dataordet i minnet vars adress finns som tredje ord i instruktionen. Resultatet skall placeras på samma adress i minnet och flaggorna skall påverkas. Operationskoden F₃₁₆ skall användas.

OR #Data,Adr RTN: M(Adr) OR Data → M(Adr), Flags → CC

OPKOD
Data
Adr

Gör en tabell liknande tabellen ovan för den efterfrågade EXECUTE-sekvensen.

(4p)

6. Besvara kortfattat följande frågor rörande FLEX-processorn.

a) I instruktionsuppsättningen finns de två ovillkorliga hoppen JMP och BRA. Vilken skillnad är det mellan dessa? Vilka fördelar och nackdelar har de? (2p)

b) Vad händer om man låter FLEX-processorn utföra en instruktion där man har glömt att sätta styrsignalen NF = 1 i sista tillståndet i "EXECUTE", om man använder styrenheten med fast logik? (2p)

c) Förklara vad som händer under processorns RESET-fas. (2p)

d) Översätt instruktionssekvensen till höger till maskinkod och visa hur den placeras i minnet. Det skall framgå hur offset för branch-instruktionerna beräknas. (3p)

e) Hur lång tid tar instruktionssekvensen i d) att köra om FLEX-processorn klockas med frekvensen 1MHz? (3p)

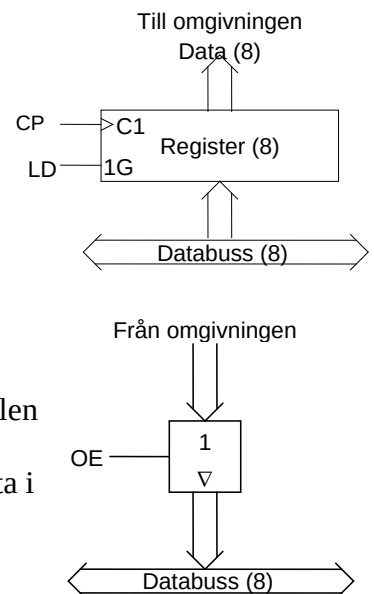
	ORG	\$20
	LDX	#TAB
	LDAB	#12
	LDA	11,X
*		
LOOP1	LDX	B,X
*		
LOOP2	DEX	
	NOP	
	BPL	LOOP2
*		
	LDX	#TAB
	INCA	
	BMI	LOOP1
*		
	BRA	NEXT
*		
TAB	RMB	8
	FCB	1,-3,5,-5,9,-10,20
*		
NEXT	NOP	

7. FLEX-datorn som visas i bilaga 3 saknar in- och utportar. Den skall nu kompletteras med en inport och en utport på adressen FC₁₆. Principen för portarna visas i figurerna till höger.

a) Konstruera grindnät som bildar signalerna LD_{FC} och OE_{FC}. Standardgrindar med valfritt antal ingångar får användas. (2p)

b) Adressen FC₁₆ är gemensam för portarna och minnet. Vilka konsekvenser får detta? (2p)

c) I ett visst program matar man ut olika datavärden på utporten från olika ställen i programmet. För att kunna mata ut "rätt" datavärde i detta speciella fall behöver man veta vad man senast matade ut på utporten. Hur löser man detta i programmet? (2p)



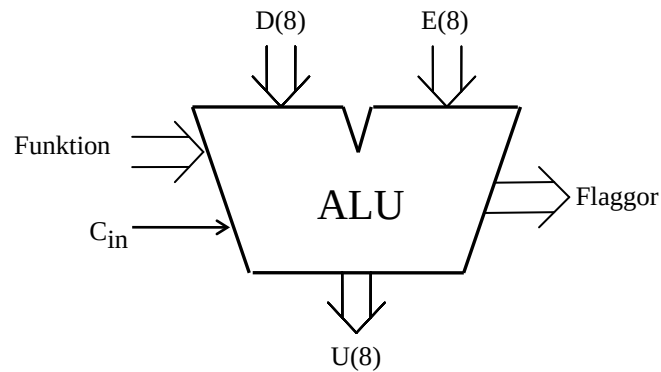
8. I minnet i ett datorsystem med FLEX-processorn finns en nollterminerad sträng med sju bitars ASCII-tecken. Varje ASCII-tecken har en paritetsbit för udda paritet som bit nr 7.

Skriv en subrutin i assemblerpråk för FLEX-processorn som tar reda på hur många ASCII-tecken för vagnretur ("CR") och radframmatning ("LF") det finns i strängen. Antalet "CR" skall returneras i A-registret och antalet "LF" i B-registret vid återhopp från subrutinen. Vid anrop av subrutinen skall startadressen till strängen finnas i X-registret.

Register A, B och CC får vara förändrade vid återhopp från subrutinen. För full poäng på uppgiften skall programmet vara korrekt radkommenterat. (6p)

Bilaga 1

ALU:ns funktion



ALU:ns **logik-** och **aritmetikoperationer** på indata **D** och **E** definieras av ingångarna **Funktion (F)** och **C_{in}** enligt tabellen nedan. **F = (f₃, f₂, f₁, f₀)**.

I kolumnen Operation förklaras hur operationen utförs.

f ₃ f ₂ f ₁ f ₀	U = f(D,E,C _{in})	
	Operation	Resultat
0 0 0 0	bitvis nollställning	0
0 0 0 1		D
0 0 1 0		E
0 0 1 1	bitvis invertering	D _{1k}
0 1 0 0	bitvis invertering	E _{1k}
0 1 0 1	bitvis OR	D OR E
0 1 1 0	bitvis AND	D AND E
0 1 1 1	bitvis XOR	D XOR E
1 0 0 0	D + 0 + C _{in}	D + C _{in}
1 0 0 1	D + FFH + C _{in}	D – 1 + C _{in}
1 0 1 0		D + E + C _{in}
1 0 1 1	D + D + C _{in}	2D + C _{in}
1 1 0 0	D + E _{1k} + C _{in}	D – E – 1 + C _{in}
1 1 0 1	bitvis nollställning	0
1 1 1 0	bitvis nollställning	0
1 1 1 1	bitvis ettställning	FFH

Carryflaggan (C) innehåller minnessiffran ut (carry-out) från den mest signifikanta bitpositionen (längst till vänster) om en aritmetisk operation utförs av ALU:n.

Vid **subtraktion** gäller för denna ALU att **C = 1 om lånesiffra (borrow) uppstår och C = 0 om lånesiffra inte uppstår**.

Carryflaggans värde är 0 vid andra operationer än aritmetiska.

Overflowflaggan (V) visar om en aritmetisk operation ger "overflow" enligt reglerna för 2-komplementaritmetik.

V-flaggans värde är 0 vid andra operationer än aritmetiska.

Zeroflaggan (Z) visar om en ALU-operation ger värdet noll som resultat på U-utgången.

Signflaggan (N) är identisk med den mest signifikanta biten (teckenbiten) av utsignalen U från ALU:n.

Half-carryflaggan (H) är minnessiffran (carry) mellan de fyra minst signifikanta och de fyra mest signifikanta bitarna i ALU:n.

H-flaggans värde är 0 vid andra operationer än aritmetiska.

I tabellen ovan avser "+" och "-" **aritmetiska operationer**. Med t ex **D_{1k}** menas att samtliga bitar i **D** inverteras.

Bilaga 2

Assemblerspråket för FLEX-processorn.

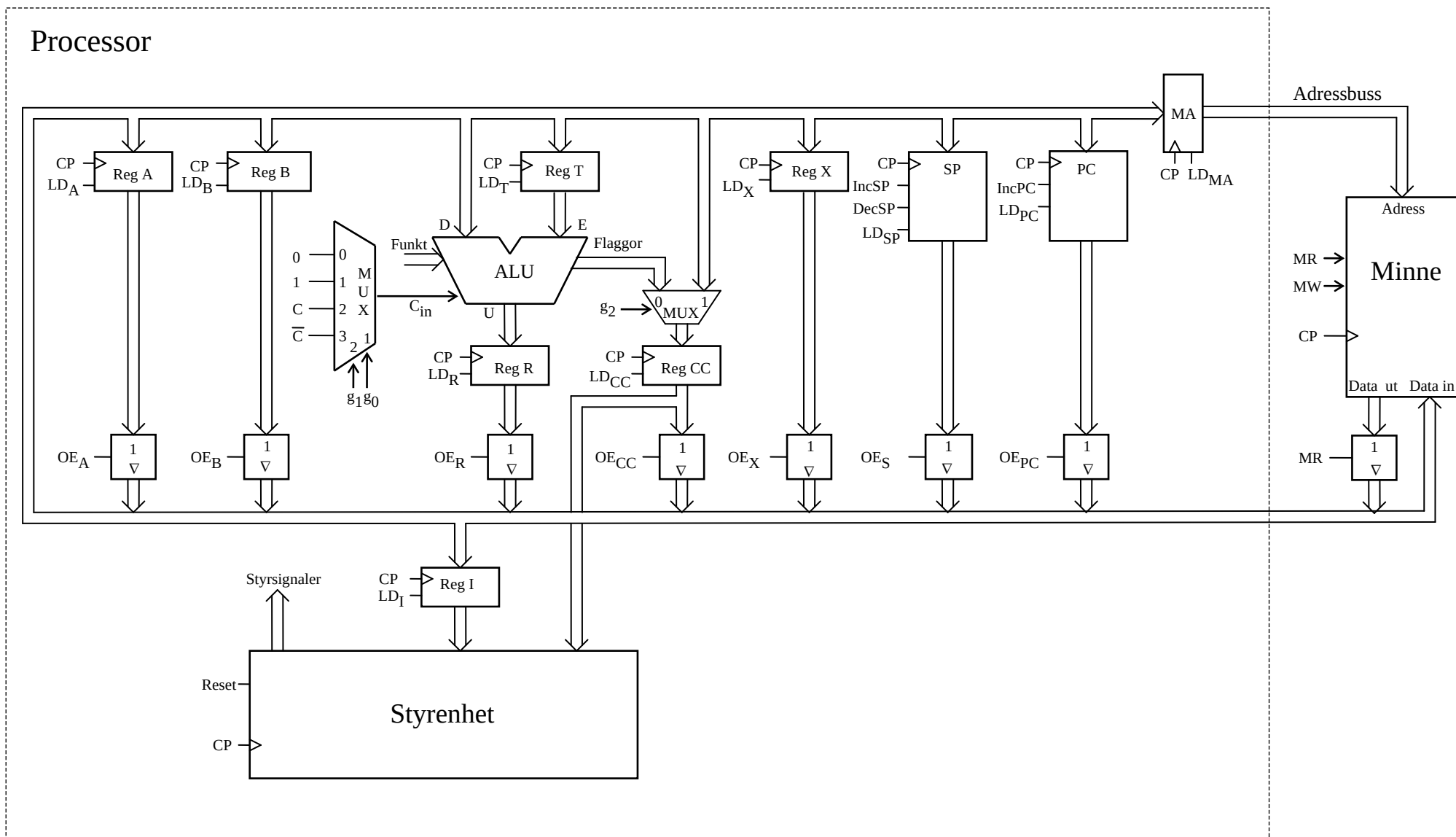
Assemblerspråket använder sig av mnemoniska beteckningar liknande dem som processorkonstruktören MOTOROLA (FREESCALE) specificerat för maskininstruktioner för mikroprocessorer 68XX och instruktioner till assemblern, såsom pseudoinstruktioner eller assemblerordirektiv. Pseudoinstruktionerna listas i tabell 1.

Tabell 1

Direktiv	Förklaring
ORG N	Placerar den efterföljande koden med början på adress N. (ORG för ORiGin = ursprung)
L RMB N	Avsätter N bytes i följd i minnet (utan att ge dem värden), så att programmet kan använda dem. Följden placeras med början på adressen L. (RMB för Reserve Memory Bytes)
L EQU N	Ger symbolen L konstantvärdet N. (EQU för EQUates = beräknas till)
L FCB N1,N2	Avsätter en byte för varje argument i följd i minnet. Respektive byte ges konstantvärdet N1, N2 etc. Följden placeras med början på adressen L. (FCB för Form Constant Byte)
L FCS "ABC"	Avsätter en byte för varje tecken i teckensträngen "ABC" i följd i minnet. Respektive byte ges ASCII-värdet för A B C, etc. Följden placeras med början på adressen L. (FCS för Form Character String)

Tabell 2 7-bitars ASCII

000	001	010	011	100	101	110	111	b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀
NUL	DLE	SP	0	@	P	`	p	0 0 0 0
SOH	DC1	!	1	A	Q	a	q	0 0 0 1
STX	DC2	"	2	B	R	b	r	0 0 1 0
ETX	DC3	#	3	C	S	c	s	0 0 1 1
EOT	DC4	\$	4	D	T	d	t	0 1 0 0
ENQ	NAK	%	5	E	U	e	u	0 1 0 1
ACK	SYN	&	6	F	V	f	v	0 1 1 0
BEL	ETB	'	7	G	W	g	w	0 1 1 1
BS	CAN	(	8	H	X	h	x	1 0 0 0
HT	EM	)	9	I	Y	i	y	1 0 0 1
LF	SUB	*	:	J	Z	j	z	1 0 1 0
VT	ESC	+	;	K	[Ä	k	{ä	1 0 1 1
FF	FS	,	<	L	\Ö	l	ö	1 1 0 0
CR	GS	-	=	M	]Å	m	}å	1 1 0 1
S0	RS	.	>	N	^	n	~	1 1 1 0
S1	US	/	?	O	_	o	RUBOUT (DEL)	1 1 1 1



Figur 1. Datorn FLEX.